Planning Mobile Robots with Boolean-based Specifications

Cristian Mahulea and Marius Kloetzer

***Abstract*— This research proposes an automated method for planning a team of mobile robots such that a Boolean-based mission is accomplished. The specification consists of logical requirements over some regions of interest for the agents' trajectories and for their final states. A Petri net with outputs models the movement capabilities of the team and the active regions of interest. The imposed specification is translated to a set of linear restrictions for some binary variables, the robot movement capabilities are formulated as linear constraints on Petri net markings, and the evaluations of the binary variables are linked with Petri net markings via linear inequalities. This allows us to solve a Mixed Integer Linear Programming problem whose solution yields robotic trajectories satisfying the task. The method is implemented as a software package and simulation results are included.**

I. INTRODUCTION

A fair amount of researches propose planning algorithms for mobile robots. The motion tasks range from classical single-robot target reachability and obstacle avoidance [1] to high-level missions for a whole team [2]. Many approaches reduce the robot interaction with the environment into finite representations, and then reason on the obtained discrete event systems [3], [4], [2], [5].

In this paper we propose the problem of planning a team of cooperating robots such that a Boolean-based specification over some regions of interest is accomplished. To this goal, we model the team movement and the satisfaction of regions with a discrete event system in form of a Petri net (PN) with outputs. Then, we convert the mission into a set of linear inequalities, we link the binary variables from these inequalities with PN markings and we obtain a Mixed Integer Linear Programming (MILP) formulation for the initial problem. The solution yields individual robot trajectories optimal from the point of view of the number of discrete transitions. The computational complexity can be lowered by reducing the PN system, at the price of obtaining suboptimal solutions. The main contributions of this work consist in the defined PN system that easily handles a whole team of agents and in the MILP formulation that includes the targeted specification together with the constructed model.

Related problems to the one we consider are reported in works as [2], [4], [6]. Although the specifications we consider here are less expressive than Linear Temporal Logic (LTL) or regular expression (as in [2]), our solution is completely different and has several advantages. Thus, instead of combining individual robot abstractions and specification automaton into a complex model, the PN model we construct has fixed topology and only the number of tokens varies with the number of robots (similar to models from [7]). Due to the assumed specification, the robots can individually follow their trajectories, without having to synchronize as it was necessary in the case of more complex tasks [2], [5]. The discrete path planning problem from [4] considers a single mobile robot instead of a team, and a task combining Boolean variables on the graph nodes. The solution translates the specification to a Traveling Salesman Problem and uses the corresponding algorithms. Work [6] presents a receding horizon framework for a single system, with specification given as LTL formulae expressed on some regions of interest. The proposed control structure uses a state space discretization of the system and a MILP problem, but the number of discrete steps to ensure the satisfaction of the formula can be large and no upper bound is available. Our method deals with a team of robots, the PN model allows the usage of some structural properties, and the number of transitions is upper-bounded. PN models have been used for modeling and controlling mobile robots in the recent literature [8], [9]. However, the modeling methodology is different, in our case the environment being partitioned depending on the regions of interest. Recently, abstractions characteristic to Resource Allocation Systems are used based on finite automata [10] or PNs [11] and the methods available for deadlock avoidance have been adapted. This work aims to obtaining a trajectory that satisfies a formula, rather than checking a given trajectory against specific properties.

The paper is structured as follows. Sec. II includes preliminaries and team model construction. An optimal solution minimizing the total number of discrete transitions is described in Sec. III, and a suboptimal but less complex adaptation is presented in Sec. IV. Sec. V includes simulation examples and Sec. VI concludes the presentation.

II. PRELIMINARIES AND TEAM MODEL

Sec. II-A defines the discrete event model that we will use for a team of identical robots. Sec. II-B introduces the formalism for expressing mission requirements for a team of cooperating robots.

A. Petri nets

This subsection introduces the basic notions of PN (see [12] for a gentle introduction).

***Definition** 2.1:* A Petri net (PN) is a tuple $\mathcal{N} = \langle P, T, F \rangle$ with P and T two finite, non-empty and disjoint sets of

C. Mahulea is with the Aragón Institute of Engineering Research (I3A), University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain {cmahulea@unizar.es}.

M. Kloetzer is with the Dept. of Automatic Control and Applied Informatics, Technical University "Gheorghe Asachi" of Iasi, Romania {kmarius@ac.tuiasi.ro}.

places and transitions; $F \subseteq (P \times T) \cup (T \times P)$ is the set of direct arcs from places to transitions or transitions to places.

The PN structure can be represented by two matrices: $\boldsymbol{Pre}, \boldsymbol{Post} \in \{0,1\}^{|P|\times|T|}$, with $\boldsymbol{Pre}[p_i,t_j] = 1$ if $\exists(p_i,t_j) \in F$, and $\boldsymbol{Pre}[p_i,t_j] = 0$ otherwise; $\boldsymbol{Post}[p_i,t_j] = 1$ if $\exists(t_j,p_i) \in F$, otherwise $\boldsymbol{Post}[p_i,t_j] = 0$. [1]

For $x \in P \cup T$, the sets of its input and output nodes (places or transitions) are denoted as ${}^{\bullet}x$ and $x^{\bullet}$, respectively. Let p_i, $i = 1,\ldots,|P|$ and t_j, $j = 1,\ldots,|T|$ denote the places and transitions. Each place can contain a non-negative integer number of tokens, and this number represents the marking of the place. The distribution of tokens in places is denoted by $\boldsymbol{m}$, where $\boldsymbol{m}[p_i]$ is the marking of place p_i. The initial token distribution, denoted by $\boldsymbol{m}_0 \in \mathbb{N}_{\geq 0}^{|P|}$, is called the initial marking of the net system. A PN with an initial marking is a PN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$.

A transition $t_j \in T$ is enabled at $\boldsymbol{m}$ if all its input places contain at least one token, i.e., $\forall p_i \in {}^{\bullet}t_j, \boldsymbol{m}[p_i] \geq 1$. An enabled transition t_j can fire leading to a new state $\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C}[\cdot,t_j]$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token flow matrix and $\boldsymbol{C}[\cdot,t_j]$ is the column corresponding to t_j. It will be said that $\tilde{\boldsymbol{m}}$ is a reachable marking that has been reached from $\boldsymbol{m}$ by firing t_j and it is writen as $\boldsymbol{m}[t_j\rangle\tilde{\boldsymbol{m}}$.

If $\tilde{\boldsymbol{m}}$ is reachable from $\boldsymbol{m}$ through a finite sequence of transitions $\sigma = t_{i_1}t_{i_2}\ldots t_{i_k}$, the following state (or fundamental) equation is satisfied:

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \tag{1}$$

where $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ is the firing count vector, i.e., its j^{th} element is the cumulative amount of firings of t_j in the sequence σ. Notice that Eq. (1) is only a necessary condition for the reachability of a marking. The marking solutions of (1) that are not reachable are called *spurious markings*. In general, checking if a marking $\boldsymbol{m}$ is reachable or not is not an easy problem due to these spurious markings.

A PN with each transition having at most one input and at most one output place is called *state machine*. Formally, a PN is state machine if $|{}^{\bullet}t| \leq 1$ and $|t^{\bullet}| \leq 1, \forall t \in T$. A PN is called live if from any reachable marking any transition can eventually fire (possibly after first firing other transitions). It is well known that for state machine PNs, liveness is equivalent to strongly connectedness and non-emptiness of (initial) marking. Moreover, in a live state machine, there exist no spurious markings [13], i.e., the solutions of the fundamental Eq. (1) give the set of reachable markings.

We will use the PN to model a a team of identical robots evolving in an environment where some convex polygonal regions of interest exist. The regions of interest are labeled with elements from set $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Pi|}\}$. For this reason, we define a class of Petri nets with outputs, which is a restrictive class of Interpreted Petri nets [14], without inputs associated to transitions.

[1]Throughout this paper, instead of using integers to refer elements of matrices or vectors, we use symbolic variables which refer the element corresponding to the used symbol.

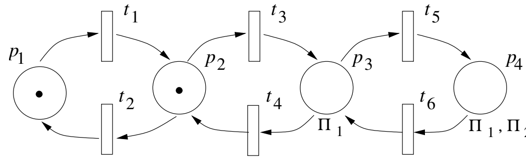

Fig. 1. A PN with outputs considered in Ex. 2.3.

Definition 2.2: A Petri net $\mathcal{Q}$ with outputs is a 4-tuple $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, \Pi, h \rangle$, where:

- $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is a Petri net system;
- $\Pi \cup \{\emptyset\}$ is the output alphabet (set containing the possible output symbols (observations)), where $\emptyset$ denotes the empty observation;
- $h : P \to 2^{\Pi}$ is an observation map, where $h(p_i)$ yields the output of place $p_i \in P$. If p_i has at least one token, then observations from $h(p_i)$ are active.

Let $\boldsymbol{v}_{\Pi_i} \in \{0,1\}^{1\times|P|}$ be the characteristic row vector of the observation $\Pi_i \in \Pi$ such that $\boldsymbol{v}_{\Pi_i}[p_k] = 1$ if $\Pi_i \in h(p_k)$ and $\boldsymbol{v}_{\Pi_i}[p_k] = 0$ otherwise. It is easy to observe that, for a reachable marking $\boldsymbol{m}$, if the product $\boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} \geq 0$ then the observation Π_i is active at $\boldsymbol{m}$. Let $\boldsymbol{V} \in \{0,1\}^{|\Pi|\times|P|}$ be the matrix formed by the characteristic vectors of all observations, i.e, the first row is the characteristic vector of Π_1, etc. The product $\boldsymbol{V} \cdot \boldsymbol{m}$ is a column vector of dimension $|\Pi|$ where the i^{th} element is non-zero if observation Π_i is active. We denote by $||\boldsymbol{V} \cdot \boldsymbol{m}||$ the set of outputs corresponding to non-zero elements of $\boldsymbol{V} \cdot \boldsymbol{m}$, i.e. $||\boldsymbol{V} \cdot \boldsymbol{m}||$ is the set of active observations (element of 2^{Π}) at marking $\boldsymbol{m}$.

Example 2.3: Let us consider the PN model from Fig. 1, which consists of $P = \{p_1,p_2,p_3,p_4\}$, $T = \{t_1,t_2,t_3,t_4,t_5,t_6\}$ and $F = \{(p_1,t_1), (t_1,p_2), (p_2,t_2), (t_2,p_1), (p_2,t_3), (t_3,p_3), (p_3,t_4), (t_4,p_2), (p_3,t_5), (t_5,p_4), (p_4,t_6), (t_6,p_3)\}$. The initial marking of the net system is $\boldsymbol{m}_0 = [1,1,0,0]^T$, the output alphabet is $\Pi = \{\Pi_1, \Pi_2\}$ and the observation map: $h(p_1) = h(p_2) = \emptyset$, $h(p_3) = \Pi_1$ and $h(p_4) = \{\Pi_1, \Pi_2\}$.

The characteristic vector of Π_1 is $\boldsymbol{v}_{\Pi_1} = [0,0,1,1]$ since Π_1 can be observed in p_3 and p_4, while $\boldsymbol{v}_{\Pi_2} = [0,0,0,1]$ since Π_2 can be observed only in p_4. Therefore, $\boldsymbol{V} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Because $\boldsymbol{V} \cdot \boldsymbol{m}_0 = [0,0]^T$, no observation is active at $\boldsymbol{m}_0$. For $\boldsymbol{m} = [0,0,2,0]^T$, $||\boldsymbol{V} \cdot \boldsymbol{m}|| = ||[2,0]^T|| = \{\Pi_1\}$, meaning that only Π_1 is active (observed) at marking $\boldsymbol{m}$. ■

A *run* (or *trajectory*) of $\mathcal{Q}$ is a *finite sequence* $r = \boldsymbol{m}_0[t_{j_1}\rangle\boldsymbol{m}_1[t_{j_2}\rangle\boldsymbol{m}_2[t_{j_3}\ldots t_{j_{|r|}}\rangle\boldsymbol{m}_{|r|}$ that induces an output *word* denoted by $h(r)$, which is the observed sequence of elements from 2^{Π}, i.e., $h(r) = ||\boldsymbol{V}\cdot\boldsymbol{m}_0||, ||\boldsymbol{V}\cdot\boldsymbol{m}_1||, \ldots, ||\boldsymbol{V}\cdot\boldsymbol{m}_{|r|}||$, $h(r) \in \left(2^{\Pi}\right)^*$, where $\left(2^{\Pi}\right)^*$ is the Kleene closure of set 2^{Π}.

The above PN with outputs can model the movement capabilities of a team of identical mobile robots in a partitioned environment cluttered with overlapping and static regions of interest denoted by elements of set Π. Such finite abstractions can be constructed based on partitions yielded by cell decompositions [15] and control laws for specific robot dynamics [16], [17]. The main idea is that the environment is partitioned based on regions of interest, every place of $\mathcal{N}$ corresponds to a partition cell, while transitions of PN correspond to robot's movement capabilities between adjacent cells. The satisfaction map h shows the regions from Π that are satisfied (visited) when the robots are inside particular cells, with empty observation corresponding to partition cells that are not included in any region from Π. The number of tokens of the PN model is equal with the number of robots, and the initial marking is given by the cells initially occupied by the team. Thus, adding a robot in the team implies adding a token to a place, without changing the PN structure.

We further assume that the model $\mathcal{Q}$ for robots evolving in an environment is already available. The informal steps that lead to its construction are captured in Alg. 1. For polygonal regions of interest, multiple cell decomposition techniques can be used in line 1 [15], our approach not being tailored for a specific one. The transitions added on lines 4-7 assume fully-actuated point robots, which can move from the current cell to any adjacent cell. For different robot dynamics, the condition from line 5 can be replaced with the existence of control laws steering the robot from cell p_i to adjacent cell p_j in finite time, e.g., works as [16], [17] describe the case of affine or multi-affine dynamics in polytopal or rectangular environments. Line 9 adds the tokens, based on robots' initial positions. The observation map from line 10 is well-defined, since the referred cell decomposition techniques preserve boundaries and intersections of regions from Π, and therefore all points inside a cell satisfy the same set of regions.

Algorithm 1: Construct the PN system $\mathcal{Q}$

1 Construct a cell decomposition of the environment based on the polygonal regions of interest from Π;
2 Associate each cell from decomposition to a place from P; let $P = \{p_1, p_2, \ldots, p_{|P|}\}$;
3 Let $T = \emptyset$, $F = \emptyset$;
4 **for** $p_i, p_j \in P,\ p_i \neq p_j$ **do**
5   **if** cells p_i and p_j are adjacent **then**
6   Add transitions $t_{i,j}$ and $t_{j,i}$ to T;
7         $F := F \cup \{(p_i, t_{i,j}), (t_{i,j}, p_j), (p_j, t_{j,i}), (t_{j,i}, p_i)\}$
8 **for** $p_i \in P$ **do**
9     $\boldsymbol{m}_0[p_i]$ = number of robots initially deployed in cell p_i;
10     $h(p_i) = \{\Pi_j \in \Pi | \text{cell } p_i \text{ is included in region } \Pi_j\}$;

***Property** 2.4:* The construction from Alg. 1 ensures that the obtained PN is a state machine.

B. Boolean-based specifications

Assume the finite set of atomic propositions $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Pi|}\}$, where in a robot-inspired scenario Π_i labels a specific region of interest from the environment.

Syntactically, we assume requirements expressed as Boolean logic formulae defined over the set of variables $\mathcal{P} = \mathcal{P}_t \cup \mathcal{P}_f$, where $\mathcal{P}_t = \Pi$ and $\mathcal{P}_f = \{\pi_1, \pi_2, \ldots, \pi_{|\Pi|}\}$, by using the standard logical connectors $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction). The sets $\mathcal{P}_t$ and $\mathcal{P}_f$ refer to the same regions of interest, but the elements of $\mathcal{P}_t$ suggest regions that should be visited along a trajectory, while $\mathcal{P}_f$ suggests regions that should be visited in the last state of a run, as explained in the below semantics.

The specifications are interpreted over finite words over the set 2^{Π}, as are those generated by the PN system with outputs $\mathcal{Q}$ from Def. 2.2. Semantically, the lower- and upper-case notations from the above set $\mathcal{P}$ have the following meaning when interpreted over the word generated by run $r = \boldsymbol{m}_0[t_{j_1}\rangle \boldsymbol{m}_1[t_{j_2}\rangle \boldsymbol{m}_2[t_{j_3} \ldots t_{j_{|r|}}\rangle \boldsymbol{m}_{|r|}$:

- $\Pi_i \in \mathcal{P}_t$ evaluates to *True* over word $h(r)$ if and only if $\exists j \in \{0, 1, \ldots, |r|\}$ such that $\Pi_i \in ||\boldsymbol{V} \cdot \boldsymbol{m}_j||$;
- $\pi_i \in \mathcal{P}_f$ evaluates to *True* over word $h(r)$ if and only if $\Pi_i \in ||\boldsymbol{V} \cdot \boldsymbol{m}_{|r|}||$.

In other words, an upper-case variable refers to a proposition that is evaluated along the whole run, while a lower-case one refers only to the final (terminal) marking. Under this explanations, the formal definitions of syntax and semantics of used specifications is not included, and it can be found in any study including Boolean formulae [18]. From now on, we will assume that any Boolean-based requirement φ is expressed into a Conjunctive Normal Form (CNF), the conversion into such a form being possible for any logical expression [18].

For example, a specification for mobile robots as $\varphi = (\Pi_1 \vee \Pi_2) \wedge \neg\pi_1 \wedge \neg\Pi_3$ requires that either region Π_1 or Π_2 is visited along the run, Π_3 is always avoided, and region Π_1 is not true (no robot occupies it) in the final state, i.e., when all robots stop.

This paper is concerned with developing supervisory control algorithms for discrete event systems (PNs) such that a Boolean-based specification is satisfied, with applications in planning a team of mobile robots. Future research will handle the expressivity of the assumed specifications when compared with other formal specifications as regular expressions or tasks capturing nonterminating behaviors. For now, we can mention that the proposed formulae are more expressive than classical reachability (navigation) tasks for mobile robots and less expressive (but easier to formally write) than regular expressions.

III. Problem definition and solution

The problem we solve is formulated as follows:

Problem: Consider a team of N identical mobile robots evolving in an environment where regions of interest labeled with elements from set Π are defined. Given a Boolean-based specification φ as in Sec. II-B, plan the motion of the robotic team such that the resulting trajectories satisfy φ.

We emphasize that specification φ imposes the requirement for the whole team of robots, instead of explicitly assigning a logical formula for each agent.

Assumptions: As stated in Sec. II-A, the team is abstracted into a PN system with outputs $\mathcal{Q}$ having the form from Def. 2.2. Under the natural assumption of a connected environment, the PN model $\mathcal{Q}$ is strongly connected (i.e. $\forall x_i, x_j \in P \cup T$ there exists a path starting in x_i and ending in x_j). Thus, the PN has no spurious markings and the set of reachable markings of the net system can be characterized by the state equation (1).

Let us assume that the requirement φ (expressed in CNF) consists of a conjunction of n terms, denoted by: $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$. Each term φ_i, $i = 1, \ldots, n$ is a disjunction of n_i variables (negated or not) from set $\mathcal{P}$ from Sec. II-B, having the form $\varphi_i = [\Pi_{j_1} \,|\, \neg\Pi_{j_1}] \vee [\pi_{j_1} \,|\, \neg\pi_{j_1}] \vee [\Pi_{j_2} \,|\, \neg\Pi_{j_2}] \vee [\pi_{j_2} \,|\, \neg\pi_{j_2}] \vee \ldots \vee [\Pi_{j_{n_i}} \,|\, \neg\Pi_{j_{n_i}}] \vee [\pi_{j_{n_i}} \,|\, \neg\pi_{j_{n_i}}]$. In the expression of φ_i, the square brackets "[...]" contain optional appearing terms, while "|" denotes a choice between two variables. For example, if $\varphi = (\Pi_1 \vee \Pi_2) \wedge \neg\pi_1 \wedge \neg\Pi_3$, then $\varphi_1 = \Pi_1 \vee \Pi_2$, $\varphi_2 = \neg\pi_1$ and $\varphi_3 = \neg\Pi_3$.

Solution main steps: Our solution begins by converting specification φ into linear restrictions over a set of $2 \cdot |\Pi|$ binary variables (Sec. III-A). Then, the values of these binary variables are linked with the PN model $\mathcal{Q}$ and a solution optimal from the point of view of the number of fired transitions is found by solving a MILP problem (Sec. III-B). The individual robot trajectories are easily obtained from the MILP solution.

A. Linear restrictions for φ

***Definition** 3.1:* Define a binary vector $\boldsymbol{x}$ with $2 \cdot |\Pi|$ variables, denoted by $\boldsymbol{x} = [x_{\Pi_1}, x_{\Pi_2}, \ldots, x_{\Pi_{|\Pi|}}, x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_{|\Pi|}}]^T \in \{0,1\}^{2\cdot|\Pi|}$, as follows:

- $x_{\Pi_i} = 1$ (or $\boldsymbol{x}[\Pi_i] = 1$) if proposition Π_i evaluates to *True* (i.e., region labeled with Π_i is visited along the team trajectory), and $x_{\Pi_i} = 0$ (or $\boldsymbol{x}[\Pi_i] = 0$) otherwise;
- $x_{\pi_i} = 1$ (or $\boldsymbol{x}[\pi_i] = 1$) if proposition π_i evaluates to *True* (i.e., a robot stops inside the region labeled with Π_i), and $x_{\pi_i} = 0$ (or $\boldsymbol{x}[\pi_i] = 0$) otherwise, $\forall i = 1, \ldots, |\Pi|$.

Under these evaluations, the satisfaction of the imposed specification φ is equivalent with a set of n linear inequalities, each such restriction corresponding to a disjunctive term φ_i, $i = 1, \ldots, n$. To formally construct these inequalities, for each φ_i, $i = 1, \ldots, n$, we define a function $\alpha_i : \mathcal{P} \to \{-1, 0, 1\}$ showing what variables from $\mathcal{P}$ appear in disjunction φ_i and which of them are negated:

$$\alpha_i(\gamma) = \begin{cases} -1, & \text{if } \neg\gamma \text{ appears in } \varphi_i \\ 0, & \text{if } \gamma \text{ does not appear in } \varphi_i \quad, \forall \gamma \in \mathcal{P} \\ 1, & \text{if } \gamma \text{ appears in } \varphi_i \end{cases} \tag{2}$$

The linear inequality corresponding to disjunction φ_i is given by:

$$\sum_{\gamma \in \mathcal{P}} (\alpha_i(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{P}} min\,(\alpha_i(\gamma), 0)\ , \tag{3}$$

where $min\,(\alpha_i(\gamma), 0)$ is the minimum value between $\alpha_i(\gamma)$ and 0.

Informally, (2) and (3) come from the following ideas: if the region corresponding to symbol $\gamma \in \mathcal{P}$ is not captured in φ_i, then its corresponding binary variable is unconstrained (it has coefficient $\alpha_i(\gamma)$ equal to zero). From all regions that appear non-negated in disjunction φ_i, at least one should be visited and thus the sum of all their corresponding binary variables should be greater or equal than 1. In (3), the non-negated symbols have coefficient 1 and they do not alter the right-hand term, since (2) evaluates to 1 for these symbols. A negated symbol γ means the avoidance of a region (either along trajectory or in final state), which implies that its corresponding binary variable x_γ should be 0. Equivalently, $1 - x_\gamma = 1$ and because x_γ is binary we can write $1 - x_\gamma \geq 1$. The first term "1" from here is placed in the right-hand term of (3) via function $min\,(\alpha_i(\gamma), 0)$.

For a better understanding we include here several examples of applying expression (3) to some disjunctions: • the inequality corresponding to π is $x_\pi \geq 1$, which can be satisfied if and only if the binary x_π has value 1, • the inequality corresponding to $\neg\pi$ is $-x_\pi \geq 0$, which can be satisfied if and only if the binary x_π has value 0, • the inequality corresponding to $\Pi_1 \vee \pi_1 \vee \neg\Pi_2$ is $x_{\Pi_1} + x_{\pi_1} - x_{\Pi_2} \geq 0$, which holds only for those binary values of $x_{\Pi_1}, x_{\pi_1}, x_{\Pi_2}$ for which the disjunction is *True*.

We conclude this subsection by saying that the CNF specification $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ is algorithmically converted (by using (3)) into a system of n linear inequalities, one for each disjunctive term. For example, the specification mentioned in Sec. II-B, $\varphi = (\Pi_1 \vee \Pi_2) \wedge \neg\pi_1 \wedge \neg\Pi_3$, translates to the following system:

$$\begin{cases} x_{\Pi_1} + x_{\Pi_2} & \geq & 1 \\ x_{\pi_1} \leq & & 0 \\ x_{\Pi_3} \leq & & 0 \end{cases} \tag{4}$$

The obtained inequalities simultaneously hold only for binary values of $\boldsymbol{x}$ for which φ evaluates to *True*, under the links given in Def. 3.1. In the remainder of this section we enforce these links between binary variables and proposition satisfactions by using markings of the PN model $\mathcal{Q}$.

B. Optimal solution

Sec. III-A shows how the specification φ is transformed to linear inequalities using Boolean variables $\boldsymbol{x}$. In this section we show that these Boolean variables can be defined by using linear inequalities based on the PN markings. For simplicity, we first handle final state requirements (formulae over $\mathcal{P}_f$), and then we present the case of trajectory requirements.

Constraints on the final state. For each observation Π_i, in Sec. III-A a binary variable x_{π_i} is introduced such that $x_{\pi_i} = 1$ if π_i evaluates to *True* at the final state. The following set of linear inequalities can be used to define the value of the variable x_{π_i} at a final reachable marking $\boldsymbol{m}$:

$$\begin{cases} N \cdot x_{\pi_i} & \geq & \boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} \\ x_{\pi_i} & \leq & \boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} \end{cases} \tag{5}$$

where N is the number of robots and $\boldsymbol{v}_{\Pi_i}$ is the characteristic row vector of observation Π_i defined in Sec. II-A. Notice that, if Π_i is *True* at final marking $\boldsymbol{m}$, then $\boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} \geq 1$ and the first equation of (5) is satisfied only if $x_{\pi_i} = 1$. On the other hand, if Π_i is *False*, then $\boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} = 0$ and the second equation of (5) is satisfied only if $x_{\pi_i} = 0$.

Example 3.2: For the model of Ex. 2.3 we impose the specification $\varphi = \pi_2$, i.e., the final marking should have at least one token in p_4 because only $h(p_4)$ includes Π_2. A binary variable x_{π_2} is introduced and the formula is satisfied if the following constraint is true: $x_{\pi_2} \geq 1$.

The final marking $\boldsymbol{m}$ at which φ should be satisfied, is (a) solution of the state equation (1), i.e., $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$ and (b) solution of (5). Therefore, in order to obtain a final marking at which the formula is satisfied, a feasible solution of the following constraints should be obtained:

$$\left\{ \begin{array}{rcl} \boldsymbol{m} & = & \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ 2 \cdot x_{\pi_2} & \geq & \boldsymbol{v}_{\Pi_2} \cdot \boldsymbol{m} \\ x_{\pi_2} & \leq & \boldsymbol{v}_{\Pi_2} \cdot \boldsymbol{m} \\ x_{\pi_2} & \geq & 1 \end{array} \right. \tag{6}$$

where $\boldsymbol{v}_{\Pi_2} = [0, 0, 0, 1]^T$. Notice that any reachable marking $\boldsymbol{m}$ having at least one token in p_4 is a solution of the previous system of inequalities. ■

When finding a solution for the proposed problem, we aim to minimize the number of transitions (robot movements) along the team trajectory. Therefore, we choose the cost function $\mathbf{1}^T \cdot \boldsymbol{\sigma}$ and we formulate the following MILP for obtaining a final marking at which the formula is satisfied:

$$\begin{array}{ll} & \min \ \mathbf{1}^T \cdot \boldsymbol{\sigma} \\ \text{s.t.} & \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \sum_{\gamma \in \mathcal{P}} \left(\alpha_i(\gamma) \cdot x_\gamma \right) \ \geq \ 1 + \sum_{\gamma \in \mathcal{P}} min \left(\alpha_i(\gamma), 0 \right), \forall \varphi_i \\ & N \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}, \forall \gamma \in \mathcal{P} \\ & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}, \forall \gamma \in \mathcal{P} \\ & \boldsymbol{m} \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}, \boldsymbol{x} \in \{0, 1\}^{|\mathcal{P}|} \end{array} \tag{7}$$

where $\boldsymbol{v}_\gamma$ is the characteristic vector of $\gamma \in \mathcal{P}$. Based on the optimal solution $\boldsymbol{\sigma}$ of (7), the robot (token) trajectories are obtained by firing the enabled transitions and by storing the sequence of places visited by each token (for more details, see [7]).

Constraints on the trajectory. In order to include constraints on the trajectory we will consider a sequence of k markings $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$ such that: $\boldsymbol{m}_1 = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}_1$, $\boldsymbol{m}_0 - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_1 \geq \mathbf{0}$; $\boldsymbol{m}_2 = \boldsymbol{m}_1 + \boldsymbol{C} \cdot \boldsymbol{\sigma}_2$, $\boldsymbol{m}_1 - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_2 \geq \mathbf{0}$; $\ldots$ Informally, these constraints enforce that between PN states $\boldsymbol{m}_{i-1}$ and $\boldsymbol{m}_i$ each token moves at most through one transition (i.e., each robot advances maximum one cell). This avoids the firing of empty cycles.

In Sec. III-A, for each proposition Π_i belonging to the specification of the trajectory, a binary variable x_{Π_i} is introduced such that $x_{\Pi_i} = 1$ if Π_i evaluates to *True* along trajectory. Because the trajectory is given by the sequence of the k intermediate markings, the set of linear inequalities used to defined the value of x_{Π_i} should consider all these intermediate markings and not only the final one as in previous case. Therefore, restrictions regarding x_{Π_i} are:

$$\left\{ \begin{array}{rcl} N \cdot (k+1) \cdot x_{\Pi_i} & \geq & \boldsymbol{v}_{\Pi_i} \cdot \left(\sum_{j=0}^k \boldsymbol{m}_j \right) \\ x_{\Pi_i} & \leq & \boldsymbol{v}_{\Pi_i} \cdot \left(\sum_{j=0}^k \boldsymbol{m}_j \right) \end{array} \right. \tag{8}$$

Solution. Putting together the number of firing transitions that has to be minimized and the constraints given by the state equation and by the specification, the following optimization problem is obtained:

$$\begin{array}{ll} & \min \ \sum_{i=1}^k \mathbf{1}^T \cdot \boldsymbol{\sigma}_i \\ \text{s.t.} & \boldsymbol{m}_i = \boldsymbol{m}_{i-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_i, i = 1, \ldots, k \\ & \boldsymbol{m}_{i-1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_i \geq 0, i = 1, \ldots, k \\ & \sum_{\gamma \in \mathcal{P}} \left(\alpha_i(\gamma) \cdot x_\gamma \right) \ \geq \ 1 + \sum_{\gamma \in \mathcal{P}} min \left(\alpha_i(\gamma), 0 \right), \forall \varphi_i \\ & N \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \forall \gamma \in \mathcal{P}_f \\ & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \forall \gamma \in \mathcal{P}_f \\ & N \cdot (k+1) \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^k \boldsymbol{m}_i \right), \forall \gamma \in \mathcal{P}_t \\ & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^k \boldsymbol{m}_i \right), \forall \gamma \in \mathcal{P}_t \\ & \boldsymbol{m}_i \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{\sigma}_i \in \mathbb{N}_{\geq 0}^{|T|}, i = 1, \ldots, k \\ & \boldsymbol{x} \in \{0, 1\}^{|\mathcal{P}|} \end{array} \tag{9}$$

The convex optimization problem (9) is a standard MILP problem [19], for which there exist complete algorithms for obtaining the optimal solution, e.g., [20]. Therefore, its solution $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_k)$ constitutes a sequence of firing count vectors for PN model $\mathcal{Q}$ and thus a solution for the problem formulated at the beginning of this section. Summing up the above details, the cost function minimizes the total number of robot transitions between cells from the partitioned environment. The constraints ensure the following: • the correct functioning of model $\mathcal{Q}$ (first two lines with constraints), • the satisfaction of formula φ through its disjunctive terms and binary variables (third constraint), • the link between binary variables corresponding to the formula and PN markings for the final requirements (constraints 4 and 5) and for the trajectory requirements (constraints 6 and 7), • feasible restrictions for unknown variables (last two constraints).

Remark 3.3: The constant k in MILP (9) is a design parameter giving the maximum number of intermediate discrete states (places) of each robot. The theoretical upper-bound of k is $|T|$, because in the worst case scenario, a robot has to once follow each transition from PN (e.g., imagine a string-like PN where the "first" and "last" places have different outputs, a robot starts from the "first" place, and the formula requires to satisfy along trajectory the output of the "last" place and to satisfy in the final state the output of the "first" one). However, in practice, much lower values of k suffice. Whenever problem (9) returns a solution, that solution is optimal (no matter the value of k), and when k is chosen too small, the problem (9) becomes unfeasible. If k is larger than needed, some intermediate firing vectors $\boldsymbol{\sigma}_i$ will result zero in solution of (9).

Remark 3.4: The sequence of firing count vectors for model $\mathcal{Q}$ obtained by solving MILP (9) can be easily projected to transition firing sequences since the PN is a live state machine. Thus, one obtains a finite trajectory (sequence of places or partition cells) for each team member. The trajectory of each robot basically satisfies a part of formula φ, such that the whole team accomplishes task φ. Because φ is a Boolean-based formula as in Sec. II-B, it cannot impose any specific order for visiting regions. Therefore, each robot can individually follow its trajectory, without synchronizing with other team members. In a real scenario, local avoidance routines can be implemented on each agent such that inter-robot collisions do not occur.

IV. Suboptimal solution

The optimal solution from Sec. III-B may exhibit a high computational complexity, especially when one chooses a large number k of intermediate steps for the trajectory. In this section we lower this complexity by reducing the size of the PN model and by solving the MILP on this reduced model.

The idea of reducing the PN $\mathcal{Q}$ (Def. 2.2) is to iteratively combine any places p_i and p_j from P that satisfy $\{p_j\} \in (p_i^\bullet)^\bullet$ and $h(p_i) = h(p_j)$ (i.e., any places that have the same output and are connected through a single transition). This reduction technique is synthesized in Alg. 2, and the reduced PN model $\tilde{\mathcal{Q}}$ has the property that its output changes when moving a token from a place to another. If one thinks at the environment partition, the reduction means that any adjacent cells that satisfy the same region(s) of interest are collapsed into a single place. In different formalisms, such a reduced system is called a quotient of the initial system, constructed with respect to equivalence classes yielded by observation map [21].

Algorithm 2: Reduce the PN model by joining places with the same output

Input: $\mathcal{Q} = \langle\langle P, T, F\rangle, \boldsymbol{m}_0, \Pi, h\rangle$
Output: $\tilde{\mathcal{Q}} = \langle\langle \tilde{P}, \tilde{T}, \tilde{F}\rangle, \tilde{\boldsymbol{m}}_0, \Pi, h\rangle$

```
1 P̃ = P; T̃ = T; F̃ = F; m̃0 = m0
2 while ∃pi, pj ∈ P̃ such that {pj} ∈ (pi•)• and h(pi) = h(pj) do
3     Let tk = pi• ∩ •pj and tl = •pi ∩ pj•
4     T̃ = T̃ \ {tk, tl}
5     F̃ = F̃ \ {(pi, tk), (tk, pj), (pj, tl), (tl, pi)}
6     m̃0[pi] = m̃0[pi] + m̃0[pj]
7     P̃ = P̃ \ {pj}
```

On the reduced PN system $\tilde{\mathcal{Q}}$ we can apply the same procedure as in section III. Notice that, for a fixed value of k, the reduced size of the model induces less variables and constrains in the MILP (9). Moreover, the upper-bound of k from Rem. 3.3 is in general significantly reduced. The solution of (9) yields a sequence of transitions/markings on the reduced PN $\tilde{\mathcal{Q}}$. In this sequence only non-empty firing vectors $\boldsymbol{\sigma}_i$ from (9) are considered (see Rem. 3.3), and we denote this sequence by $\tilde{r} = \tilde{\boldsymbol{m}}_0[\tilde{t}_{j_1}\rangle\tilde{\boldsymbol{m}}_1[\tilde{t}_{j_2}\rangle\tilde{\boldsymbol{m}}_2[\tilde{t}_{j_3}\ldots\tilde{t}_{j_k}\rangle\tilde{\boldsymbol{m}}_{\tilde{k}}$, where $\tilde{k} \leq k$ and $\tilde{\boldsymbol{m}}_i \neq \tilde{\boldsymbol{m}}_{i+1}$, $i = 0, 1, \ldots, \tilde{k}-1$.

The solution $\tilde{r}$ basically shows how the observations from 2^Π should be changed such that φ is *True*, but it does not give agent trajectories as in the case of full system $\mathcal{Q}$. The firing of a single transition in $\tilde{\mathcal{Q}}$ corresponds to the firing of a sequence of transitions in the original $\mathcal{Q}$. Therefore, we need to project $\tilde{r}$ to a sequence on the original PN model to obtain the robot motions. This projection is always possible, because the construction from Alg. 2 guarantees that the team can produce the sequence of outputs from $\tilde{r}$, although some outputs are repeated in $\mathcal{Q}$. This repetitions do not affect the satisfiability of Boolean-based φ [18].

The procedure to project the solution is iterative. We show how the first sequence corresponding to $\tilde{\boldsymbol{m}}_0[\tilde{t}_{j_1}\rangle\tilde{\boldsymbol{m}}_1$ is obtained. An linear programming problem (LPP) is solved in order to obtain a firing sequence in $\mathcal{Q}$ corresponding to $\tilde{t}_{j_1}$ from $\tilde{\mathcal{Q}}$:

- Remove from $\mathcal{Q}$ all places (together with input and output transitions and corresponding arcs) having outputs different than the ones in $\tilde{\boldsymbol{m}}_0$ and $\tilde{\boldsymbol{m}}_1$. Formally, a place $p \in P$ is removed if $h(p) \neq ||\tilde{\boldsymbol{V}} \cdot \tilde{\boldsymbol{m}}_0||$ or $h(p) \neq ||\tilde{\boldsymbol{V}} \cdot \tilde{\boldsymbol{m}}_1||$, where $\tilde{\boldsymbol{V}}$ is the matrix of characteristic vectors of $\tilde{\mathcal{Q}}$ (Sec. II-A). Let $\langle\bar{\mathcal{N}}, \bar{\boldsymbol{m}}_{\boldsymbol{0}}\rangle$ be the resulted PN system. The removal of places and transitions ensures that no other output (that could violate the formula) is observed during the trajectory;
- Remove from $\langle\bar{\mathcal{N}}, \bar{\boldsymbol{m}}_{\boldsymbol{0}}\rangle$ all the strongly connected components that do not have any token in $\tilde{\boldsymbol{m}}_0$ (no transitions can be fired in such components). Thus, $\bar{\mathcal{N}}$ now contains only live strongly connected components;
- The first LPP constraint is the state equation of $\langle\bar{\mathcal{N}}, \bar{\boldsymbol{m}}_{\boldsymbol{0}}\rangle$: $\bar{\boldsymbol{m}}_f = \bar{\boldsymbol{m}}_0 + \bar{\boldsymbol{C}} \cdot \boldsymbol{\sigma}$;
- The second constraint ensures that the output at $\bar{\boldsymbol{m}}_f$ is the same as the one at $\tilde{\boldsymbol{m}}_1$, i.e.: $\bar{\boldsymbol{V}} \cdot \bar{\boldsymbol{m}}_f = \tilde{\boldsymbol{V}} \cdot \tilde{\boldsymbol{m}}_1$, where $\bar{\boldsymbol{V}}$ is the matrix formed by characteristic vectors of $\bar{\mathcal{N}}$;
- Solve the LPP minimizing the cost function $\mathbf{1}^T \cdot \boldsymbol{\sigma}$. Since $\bar{\mathcal{N}}$ is a state machine composed by live strongly connected components, a LPP solved with Simplex method is guaranteed to return a feasible integer solution [13]. The solution gives the firing sequence on the original system $\mathcal{Q}$ (hence the runs for robots) and the marking $\bar{\boldsymbol{m}}_f$ of $\mathcal{Q}$ corresponding to $\tilde{\boldsymbol{m}}_1$ of $\tilde{\mathcal{Q}}$.

The previous procedure is repeated for the second step of $\tilde{r}$ by taking $\bar{\boldsymbol{m}}_f$ as initial marking. Thus, the projection of $\tilde{r}$ to a solution of $\mathcal{Q}$ is done by solving at most k LPPs.

Overall, the procedure from this section requires the reduction from Alg. 2, a MILP problem with fewer variables and constraints than the one from (9), and a number of $\tilde{k} \leq k$ LPP problems on reduced systems of type $\bar{\mathcal{N}}$. In all the simulations we performed, this procedure required less computation time than the one from Sec. III-B. However, the reduced MILP and the local minimization from the $\tilde{k}$ LPPs do not guarantee the optimality of the solution in $\mathcal{Q}$, as was the case in Sec. III-B. This is because $\tilde{\mathcal{Q}}$ *looses* the number of transitions of $\mathcal{Q}$ that should fire such that a desired output is obtained. Examples illustrating these aspects are included in the next section.

V. Simulation examples

This section illustrates the usage of our method for planning a team of mobile robots. The described approach was implemented in Matlab as a user-friendly package available at [22]. Our implementation includes the external MILP solver from [20].

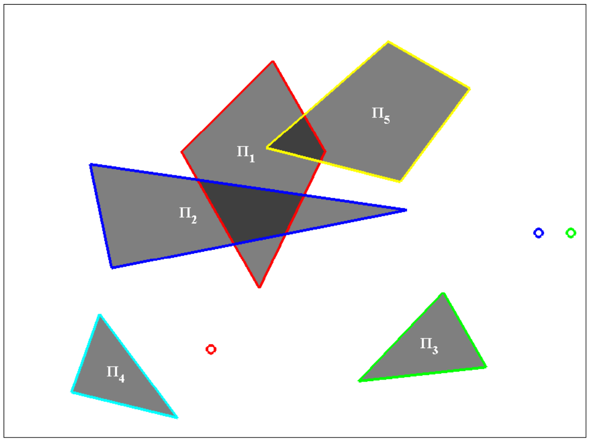

Fig. 2. A rectangular environment with five regions of interest labeled with elements of set $\Pi = \{\Pi_1, \Pi_2, \dots, \Pi_5\}$. Each region has differently colored borders, for easier understanding their overlapping. The team consists of three robots, with initial positions marked by the red, blue and green circles.

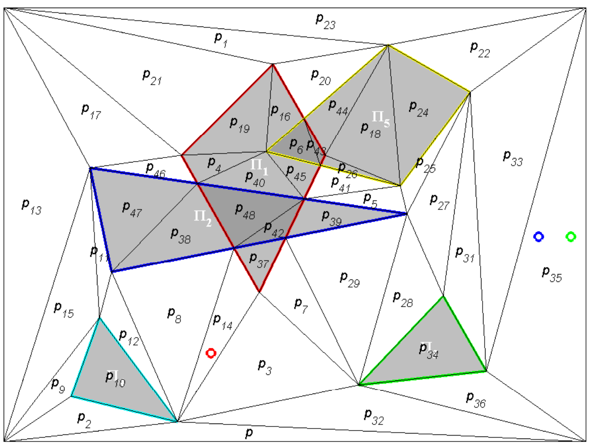

Fig. 3. Triangular partition of the environment from Fig. 2. There are 48 non-overlapping cells with the properties that every two adjacent cells exactly share one facet, and all points inside a cell belong to the same region(s) from set Π.

We consider the environment depicted in Fig. 2, where five polygonal regions are defined. For simplicity of constructing the team model, we consider $N = 3$ point and fully-actuated agents, whose initial positions are marked with circles in Fig. 2. Alg. 1 from Sec. II-A yields the PN system $\mathcal{Q}$ as follows. The environment is partitioned by using a constrained triangular decomposition [23], based on polygonal regions Π. The resulting partition has 48 cells (labeled with elements of set $P = \{p_1, p_2, \dots, p_{48}\}$) and it is shown in Fig. 3. There result 140 transitions in T, given by adjacency between cells (two triangles are adjacent if they share an entire facet). The observation map h is easily created based on the inclusion of each cell in some regions of interest, e.g., $h(p_3) = \emptyset$, $h(p_{10}) = \Pi_4$, $h(p_{48}) = \{\Pi_1, \Pi_2\}$. System $\mathcal{Q}$ has three tokens and the initial marking is given by initial team deployment: $\boldsymbol{m}_0[p_{14}] = 1$, $\boldsymbol{m}_0[p_{35}] = 1$, $\boldsymbol{m}_0[p_i] = 0$, $\forall i \in \{1, \dots, 48\}$, $i \neq 14, 35$.

Considering the syntax and semantics explained in Sec.

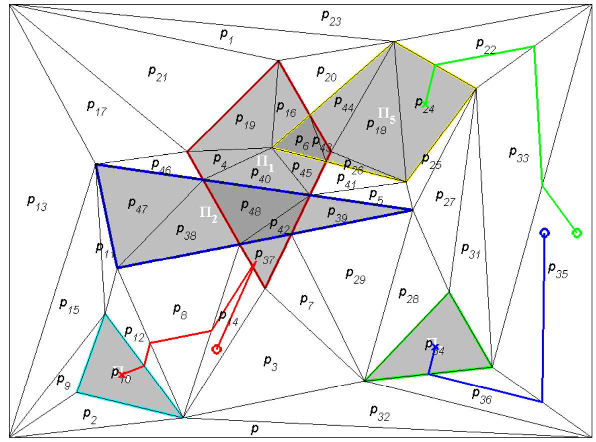

Fig. 4. Optimal solution (with respect to the overall number of transitions) comprises a total number of 10 movements between cells. Each robot follows its trajectory and stops in the point marked with "x", and thus the team fulfills mission φ.

II-B, the team mission is given by the specification:

$$\varphi = \neg\Pi_2 \wedge \Pi_1 \wedge \neg\pi_1 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5. \tag{10}$$

In words, the second region should be avoided, the first region should be visited along run, but no robot should finally remain inside it, and the last three regions should be occupied when the robots stop.

Formula φ is converted into a system of 6 linear inequalities with 6 binary variables (Sec. III-A). By adopting the optimal solution described in Sec. III-B with a maximum number of steps $k = 10$, the firing sequences translate to the following runs for the robots:

$$\begin{aligned} \text{red robot: } & p_{14}, p_{37}, p_{14}, p_8, p_{12}, p_{10} \\ \text{blue robot: } & p_{35}, p_{36}, p_{34} \\ \text{green robot: } & p_{35}, p_{33}, p_{22}, p_{24} \end{aligned} \tag{11}$$

The MILP problem from Sec. III-B was constructed in 0.5 seconds, and it includes 1890 variables (from which 1400 are integer and 10 binary), 480 equality constraints and 506 inequality constraints. The solution was obtained in around 1 second on a medium performance laptop. Under the same conditions, if k were set to 20, the running time increases to 12 minutes, from which the MILP construction took only 3 seconds.

The actual robotic trajectories are presented in Fig. 4, and they were constructed by connecting the middle points of the common edges shared by successive cells from each robot's path, this being a fast method for constructing continuous trajectories for fully-actuated robots evolving in partitioned environment with convex cells [1]. Finally, each robot converges to the centroid of the last visited cell.

If one applies the suboptimal solution from Sec. IV, the reduced PN model $\tilde{\mathcal{Q}}$ has 10 places and 26 transitions. The MILP problem construction and solving took only 0.6 seconds if k is set equal to its maximum possible value, i.e., $k = 26$, while the projection to individual robot trajectories was done in approximately 1 second. For $k = 10$, the suboptimal solution was obtained in 1.2 seconds, which

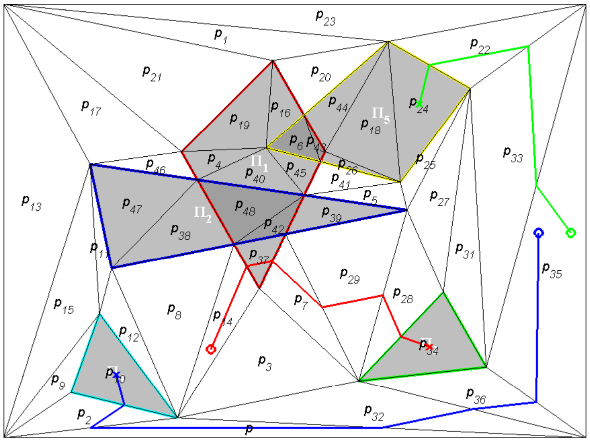

Fig. 5. Sub-optimal solution imposes 13 transitions between cells. Each robot follows its trajectory and stops in the point marked with "x", and the formula from (10) is satisfied.

shows a computationally tractable algorithm even for large problems, when compared to the optimal method. The sub-optimal solution yielded the robot paths from (12), which include a total number of 13 transitions among cells. The corresponding trajectories are shown in Fig. 5.

$$\begin{aligned} \text{red robot:} \quad & p_{14}, p_{37}, p_7, p_{29}, p_{28}, p_{34} \\ \text{blue robot:} \quad & p_{35}, p_{36}, p_{32}, p_{30}, p_2, p_{10} \\ \text{green robot:} \quad & p_{35}, p_{33}, p_{22}, p_{24} \end{aligned} \tag{12}$$

VI. Conclusions

We presented an approach that automatically plans a team of cooperating mobile robots based on a Boolean-based task given over a set of regions in the environment. The solution relies on solving a MILP optimization problem that is formulated over a discrete event system. Based on a partition of the environment, the robotic team is abstracted to a PN with outputs, which has the advantage that the topology remains fixed and only the number of tokens varies with the team size. The Boolean formula is represented through a set of linear inequalities in some binary variables, the evaluations of these variables are linked with a finite sequence of PN markings and the PN's fundamental equation is used for making sure that any obtained marking is reachable through a firing sequence. Thus, we obtain a MILP formulation for the proposed problem, and its solution provides a set of firing PN transitions which are easily converted to individual robotic trajectories. The solution is optimal with respect to the number of discrete transitions followed by team members. The computational burden can be reduced by using a suboptimal adaptation that solves the problem on a reduced PN system. Due to the considered specifications, the robots can follow their trajectories without synchronizing with other team members. We implemented our procedure as a freely-downloadable Matlab software package whose usefulness is illustrated through included simulation results. Future work will be conducted towards relating and extending the assumed tasks to other specification formalisms that may be of interest.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Boston: MIT Press, 2005.

[2] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[3] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[4] F. Imeson and S. L. Smith, "A language for robot path planning in discrete environments: The tsp with boolean satisfiability constraints," in *Proceedings of the IEEE Conf. on Robotics and Automation*, May 2014, to appear.

[5] M. Kloetzer, X. C. Ding, and C. Belta, "Multi-robot deployment from LTL specifications with reduced communication," in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011, pp. 4867–4872.

[6] T. Wongpiromsarn, U. Topcu, and R.-M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[7] M. Kloetzer and C. Mahulea, "A Petri net based approach for multi-robot path planning," *Discrete Event Dynamic Systems*, 2013, in press.

[8] H. Costelha and P. Lima, "Robot task plan representation by Petri nets: modelling, identification, analysis and execution," *Journal of Autonomous Robots*, pp. 1–24, 2012.

[9] J. King, R. Pretty, and R. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 5, pp. 615–619, 2003.

[10] S. Reveliotis and E. Roszkowska, "Conflict Resolution in Free-Ranging Multivehicle Systems: A Resource Allocation Paradigm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[11] M. Kloetzer, C. Mahulea, and J.-M. Colom, "Petri net approach for deadlock prevention in robot planning," in *Proc. of ETFA 2013: IEEE 18th Conf. on Emerging Technologies Factory Automation*, Sept 2013.

[12] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[13] M. Silva, E. Teruel, and J.-M. Colom, "Linear algebraic and linear programming techniques for the analysis of place/transition net systems." *Lecture on Petri Nets I: Basic Models*, vol. 1491, pp. 309–373, 1998.

[14] A. Ramirez-Trevino, I. Rivera-Rangel, and E. Lpez-Mellado, "Observability of discrete event systems modeled by interpreted Petri nets," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 557–565, August 2003.

[15] M. D. Berg, O. Cheong, and M. van Kreveld, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, 2008.

[16] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, pp. 938–948, 2006.

[17] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.

[18] F. Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd ed. Dover Publications, 2012.

[19] J. Chinneck, *Practical Optimization: A Gentle Introduction*. http://www.sce.carleton.ca/faculty/chinneck/po.html, 2004.

[20] A. Makhorin, "GNU linear programming kit," http://www.gnu.org/software/glpk/, 2012.

[21] R. Milner, *Communication and concurrency*. Prentice-Hall, 1989.

[22] C. Mahulea and M. Kloetzer, "Software tool for motion planning based on Boolean specifications and PN abstractions," http://webdiis.unizar.es/~cmahulea/research/PN_Boole_plan.zip, 2014.

[23] J. R. Shewchuk, "General-dimensional constrained delaunay and constrained regular triangulations, i: Combinatorial properties," *Discrete & Computational Geometry*, vol. 39, pp. 580–637, 2008.